Handling Ordinary Files

Unit 2

cat: Displaying and Creating Files

- *cat* is one of the most frequently used commands on Unix-like operating systems.
- It has three related functions with regard to text files: displaying them, combining copies of them and creating new ones.
- cat's general syntax is

  cat [options] [filenames] [-] [filenames]

cat: Displaying and Creating Files

- **Reading Files**
  - The most common use of cat is to read the contents of files, and cat is often the most convenient program for this purpose.
  - All that is necessary to open a text file for viewing on the display monitor is to type the word *cat* followed by a space and the name of the file and then press the ENTER key.
  - For example, the following will display the contents of a file named *file1*:

```
cat file1
```

cat: Displaying and Creating Files

- The standard output stream for the cat command is the monitor.
- However this can be redirected to other stream like file.
- In the following example, the standard output of cat is redirected using the *output redirection operator* (which is represented by a rightward pointing angular bracket) to *file2*:

```
cat file1 > file2
```

- That is, the output from cat is written to file2 instead of being displayed on the monitor screen.

cat: Displaying and Creating Files

- The standard output could instead be redirected using a *pipe* (represented by a vertical bar) to a *filter* (i.e., a program that transforms data in some meaningful way) for further processing.
- For example, if the file is too large for all of the text to fit on the monitor screen simultaneously, as is frequently the case, the text will scroll down the screen at high speed and be very difficult to read.

cat: Displaying and Creating Files

- This problem is easily solved by *piping* the output to the filter *less*, i.e.,

  cat file1 | less
- This allows the user to advance the contents of the file one screenful at a time by pressing the space bar and to move backwards by pressing the *b* key. The user can exit from less by pressing the *q* key.

cat: Options (-v and –n)

- There are two cat options that you may find useful;
  - – Displaying Non Printing Characters
    - Cat is normally used for displaying text files only.
    - Executables , when seen with cat, simply display junk.
    - If you have non printing characters in your input , you can use cat with the –v option to display the characters.

cat: Options (-v and –n)

– Numbering Lines (-n)
  - The –n option is used to display numbered lines.
  - It is necessary if you need to debug a program and locate an error on a line number.

Using cat to create a file

- The second use for cat is file creation.
- For small files this is often easier than using *vi*, *gedit* or other text editors.
- It is accomplished by typing *cat* followed by the output redirection operator and the name of the file to be created, then pressing ENTER and finally simultaneously pressing the CONTROL and *d* keys.
- For example, a new file named *file1* can be created by typing

```
cat > file1
```

Using cat to create a file

- If a file named *file1* already exists, it will be *overwritten* (i.e., all of its contents will be erased) by the new, empty file with the same name.
- Thus the cautious user might prefer to instead use the *append operator* (represented by two successive rightward pointing angular brackets) in order to prevent unintended erasure. That is,

```
cat >> file1
```

then pressing the ENTER key and finally simultaneously pressing the CONTROL and *d* keys.

Using cat to create a file

- That is, if an attempt is made to create a file by using cat and the append operator, and the new file has the same name as an existing file, the existing file is, in fact, preserved rather than overwritten, and any new text is added to the end of the existing file

Concatenation of Files

- Sometimes you need to concatenate two files especially if you are following the modular coding techniques.
- You can easily concatenate two files by using the redirecting output parameter ’>’.
- We can concatenate two files into the third file by following a syntax like

```
cat file1 file2 >file3
```

- It will first write down the file1 and then file2 into the file3.

cp: Copying a file

- Use the cp command to create a copy of the contents of the file or directory specified by the *SourceFile* or *SourceDirectory* parameters into the file or directory specified by the *TargetFile* or *TargetDirectory* parameters.
- If the file specified as the *TargetFile* exists, the copy writes over the original contents of the file without warning. If you are copying more than one *SourceFile*, the target must be a directory.

cp: Copying a File

- Synopsis:

  cp [*OPTION*]... *SOURCE* *DEST*INATION

- EXAMPLE:
  - To copy a file in your current directory into another directory, type the following:

    cp jones /home/nick/clients

  This copies the jones file to /home/nick/clients/jones.

cp:Copying a File

- **Example:**
  - To make a copy of a file in the current directory, type the following:

    ```
    cp prog.c prog.bak
    ```
  - This copies prog.c to prog.bak. If the prog.bak file does not already exist, then the cp command creates it.
  - If it does exist, then the cp command replaces it with a copy of the prog.c file.

cp: Copying a File

- cp is often used with shorthand notation, .(dot), to signify the current directory as the destination
- For instance , to copy the file .profile from /home/sharma to your current directory you can use either of the two commands

```
cp  /home/sharma/.profile     .profile
cp  /home/sharma/.profile     .
```

cp Options

- Interactive Copying (-i)
  - The –I (interactive) option wars the user before overwriting the destination file.
  - If until exists, cp prompts for a response.
  - Example:

```
$ cp -I chap01 unit1
```

rm: Deleting Files

- The rm command deletes one or more files.
- It normally operates silently and should be used with caution.
- The following command deletes three files:

```
rm chap01 chap02 chap03
```

- You may need to delete all the files in a directory as a part of clean up operation.

rm: Deleting Files

- The * , when used itself , represents all files, and you can use it with rm command.

```
rm *
```

rm Options

- Interactive Deletion(-i):

  Like in cp, the –I (interactive) option makes the command ask the user for confirmation before removing each file.

- Example:

```
rm –I chap01 chap02 chap03
```

rm Options

- Force Removal (-f)

  rm prompts for removal if a file is write-protected. The –f option overrides this minr protection and forces removal.

  ```
  rm –f chap01
  ```

- If you combine this with –r option it could be more risky thing to do.

  rm –rf *   will delete everything in the current directory and below.

mv: Renaming Files

- The mv command returns (moves) files. It has two distinct functions:
  - It renames a file (or directory)
  - It moves a group of files to a different directory.
- mv doesn’t create a copy of the file; it merely renames it.
- No additional space is consumed on disk during renaming

mv: Renaming Files

mv [-f] [-i] oldname newname

-f	mv will move the file(s) without prompting even if it is writing over an existing target. Note that this is the default if the standard input is not a terminal.
-i	Prompts before overwriting another file.
oldname	The oldname of the file renaming.
newname	The newname of the file renaming.
filename	The name of the file you want to move directory - The directory of were you want the file to go.

more: Paging Output

- more is what we call a pager utility.
- Oftentimes the output of a particular command is too big to fit on one screen.
- The individual commands do not know how to fit their output to separate screens.
- They leave this job to the pager utility.
- The more command breaks the output into individual screens and waits for you to press the space bar before continuing on to the next screen.

more: Paging Output

- Pressing the enter key will advance the output one line.
- Here is a good example:

```
% cd /usr/bin %
ls –l
```

- That should scroll for a while. To break up the output screen by screen, just pipe it through more:

```
% ls -l | more
```

- The pipe is short for saying take the output of ls and feed it into more

Navigation

- Irrespective of version , more uses the space bar to scroll forward a page at a time.
- To move forward one page, use

  f or the spacebar

  and to move back one page, use

  b

Searching for a pattern

- You can perform search for a pattern with / command followed by the string.
- For instance, to look for the first while loop in your program ,

```
/while
```

- You can repeate this search for viewing the next while loop section by pressing n.
- Move back with b to arrive to the first page.

The lp subsystem: Printing a File

- No user is allowed to direct access to the printer.
- Instead one has to spool(line up) a job along with others in a print queue.
- Spooling ensures the orderly printing the jobs and relieves the user from the necessity of administering the print resources.
- The spooling facility is provided by the lp(line printing) command.

lp Options

- If printer has been defined the lp runs fine.
- If the printer is not defined or if more than one printers are defined, you need to use the -d with the printer name.

  lp –dlaser chap01.ps
- The –t option followed by the title string, prints the title on the first page.

  lp –t “First Chapter” chap01.ps

lp Options

- After the file has been printed, you can notify the user with the –m (mail) option.
- You can also print multiple copies(-n)
- Example:

```
lp –n3 –m chap01.ps
```

Other commands in the lp subsystem

- The print queue is viewed with the lpstat command.
- By viewing this list, you can use the cancel command to cancel any jobs submitted to you.
- Cancel uses the request-id or printer name as argument.
- Example:

```
cancel laser
cancel pr1-320
```

file: Knowing the File Types

- Even if we know that there are 3 types of unix files we often want to know more about these files.
- For instance a regular file may contain plain text, a c file may contain executable code.
- Unix provides the file command to determine the type of file , especially of an ordinary file.

file: Knowing the File Types

- Example:

```
file archive.zip
```

- File correctly identifies the basic file types.
- Use * to signify all the files.
- Example:

```
file *
```

wc: Counting Lines, words and characters

- Unix features a universal word-counting program that also counts lines and characters.
- It takes one or more filenames as arguments and displays a four columnar output
- Synopsis:

  wc filename

wc: Counting Lines, words and characters

- There are 3 Options with **wc**
- **-l** Counts the number of lines
- **-w** Counts the number of words
- **-m or -c** Counts the number of characters
- The -m option is available on Solaris and HP-UX. It is not available on Linux. On Linux systems, you need to use the -c option instead.

wc: Counting Lines, words and characters

- **Number of Lines**
- To count the number of lines, use the -l (*l* as in *lines*) option. For example, the command

```
$ wc -l .profile
```

- produces the output

```
133 .profile
```

wc: Counting Lines, words and characters

- **Number of Words**
- To count the number of words in a file, use the -w (*w* as in *words*) option. For example, the command

```
$ wc -w .rhosts
```

- produces the output

```
14 .rhosts
```

- which is what you expected.

wc: Counting Lines, words and characters

- **Number of Characters**
- To count the number of characters, use either the -m option or the -c option. As mentioned, the -m option should be used on Solaris and HP-UX. The -c option should be used on Linux systems.
- For example, the command

```
$ wc -c .profile
```

- produces the output

```
2908 .profile
```

od: Displaying Data in Octal

- ***od*** - dump files in octal and other formats
- ***octal dump*** displays contents as octal numbers.
- This can be useful when the output contains non-printable characters.
- For example, a filename may contain non-printable characters.
- It can also be handy to view binary files.

od: Displaying Data in Octal

- Synopsis

  od [-bc] filename
- The –b option displays this value for each character seperately.
- Example:

  od –b file1
- The –c command displays the associated character value.
- Example:

  od –bc file1

cmp: Comparing two files

- You may often need to know whether two files are identical so one of them can be deleted.
- There are three commands in the UNIX system that can tell you this.
- The cmp command compares two files byte by byte and returns the location of the first mismatch on the screen.

cmp: Comparing two files

- It takes two arguments:
- Synopsis:

```
cmp chap01 chap02
```

- Output:

```
chap01 chap02 differ: char 9, line 1
```

- If two file are identical , cmp display no message, but simply returns to the prompt.

comm: What is Common?

- The **comm** command in the Unix family of computer operating systems is a utility that is used to compare two files for common and distinct lines.
- comm reads two files as input, regarded as lines of text.
- comm outputs one file, which contains three columns.
- The first two columns contain lines unique to the first and second file, respectively

comm: What is Common?

- . The last column contains lines common to both.
- This functionally is similar to diff.
- Synopsis:

```
comm file1 file2
```

Example

File foo

```
apple
banana
eggplant
```

File bar

```
apple
banana
banana
zucchini
```

```
comm foo bar
                  apple
                  banana
          banana
eggplant
          zucchini
```

diff: Converting one file to other

- In computing, **diff** is a file comparison utility that outputs the differences between two files.
- It is typically used to show the changes between one version of a file and a former version of the same file.
- diff displays the changes made per line for text files.

diff: Converting one file to other

- It is invoked from the command line with the names of two files: diff *original new*.
- The output of the command represents the changes required to make the *original* file become the *new* file.
- If *original* and *new* are directories, then diff will be run on each file that exists in both directories.

diff: Converting one file to other

- An option, -r, will recursively descend any matching subdirectories to compare files between directories.
- Synopsis:

  diff original_file   new_file

diff: Converting one file to other

- **Usage:**
  - Suppose you have two sequences of items:
  - a b c d f g h j q z
  - a b c d e f g i j k r x y z
  - From a longest common subsequence it's only a small step to get diff-like output: if an item is absent in the subsequence but present in the original, it must have been deleted. (The '–' marks, below.)

diff: Converting one file to other

- If it is absent in the subsequence but present in the second sequence, it must have been added in. (The '+' marks.)

```
e  h i  q  k r x y
+  - +  -  + + + +
```

unix2dos and dos2unix: Converting between DOS and UNIX

- **unix2dos** (sometimes named todos or u2d) is a Unix tool to convert an ASCII text file from Unix format (line break) to DOS format (carriage return and line break) and vice versa.
- When invoked as **unix2dos** the program will convert a Unix text file to DOS format, when invoked as **dos2unix** it will convert a DOS text file to UNIX format.

unix2dos and dos2unix: Converting between DOS and UNIX

- Syopsis

```
unix2dos [-p] [file]
```

  - **p** Access and modification time of the original file are preserved.
  - **file** File to convert

Compressing and Archiving Files

- To conserve disk space you need to compress large and infrequent used files.
- Unix comes with some compressing and decompressing utilitites
  - gzip and gunzip(.gz)
  - bzip2 and bunzip2 (.bz2)
  - zip and unzip (.zip)
- Apart from compressing, you'll need to group a set of files into a single file, called archive,

Compressing and Archiving Files

- The tar and zip commands can pack an entire directory structure into an archive.
- You can send this archive as a single file , either using ftp or as an email attachment to a remote machine.

gzip and gunzip: Compressing and decompressing files

- *Gzip* reduces the size of the named files using Lempel-Ziv coding (LZ77).
- Whenever possible, each file is replaced by one with the extension **.gz,** while keeping the same ownership modes, access and modification times.
- *Gzip* will only attempt to compress regular files.
- In particular, it will ignore symbolic links.

gzip and gunzip: Compressing and decompressing files

- If the compressed file name is too long for its file system, *gzip* truncates it.
- *Gzip* attempts to truncate only the parts of the file name longer than 3 characters.
- **Synopsis:**

  gzip [*name ...*]

  Example:

  gzip hello.html

gzip and gunzip: Compressing and decompressing files

- To know the how much compression is applied on a file use –l option.
- Example:

```
gzip –l hello.html.gz
```

gzip and gunzip: Compressing and decompressing files

- *gunzip* can currently decompress files created by *gzip, zip, compress, compress -H* or *pack.*
- The detection of the input format is automatic.
- When using the first two formats, *gunzip* checks a 32 bit CRC.
- For *pack, gunzip* checks the uncompressed length.
- The standard *compress* format was not designed to allow consistency checks. However *gunzip* is sometimes able to detect a bad .gz file.

gzip and gunzip: Compressing and decompressing files

- Synopsis:

  gunzip [*name ...*]

- Example:

  gunzip hello.html.gz

  or

- gzip –d hello.html.gz

tar: The archival program

- **tar** ,is archiving program designed to store and extract files from an archive file known as a *tarfile.*
- A *tarfile* may be made on a tape drive, however, it is also common to write a *tarfile* to a normal file.
- The first argument to **tar** must be one of the options: **Acdrtux**, followed by any optional functions.
- The final arguments to **tar** are the names of the files or directories which should be archived.
- The use of a directory name always implies that the subdirectories below should be included in the archive.

tar: The archival program

- **A, --catenate, --concatenate** append tar files to an archive
- **-c, --create** create a new archive

- **-d, --diff, --compare** find differences between archive and file system
- **--delete** delete from the archive (not for use on mag tapes!)

tar: The archival program

- **-r, --append** append files to the end of an archive
- **-t, --list** list the contents of an archive
- **-u, --update** only append files that are newer than copy in archive
- **-x, --extract, --get** extract files from an archive

zip and unzip Commands

- UNIX operating system provides the following two commands:
- **zip** - Command to package and compress (archive) files.
- **unzip** - Command to list, test and extract compressed files in a ZIP archive.

zip and unzip Commands

- To create a zip file, enter:

  zip filename.zip input1.txt input2.txt resume.doc pic1.jpg
- To decompress a zip file in Unix, enter:
- **unzip** filename
- For example: **unzip** filename.**zip**